



CIC MegaCore Function

User Guide



101 Innovation Drive
San Jose, CA 95134
www.altera.com

Software Version: 11.0
Document Date: May 2011

Copyright © 2011 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Chapter 1. About This MegaCore Function

Features	1-1
Release Information	1-2
Device Family Support	1-2
MegaCore Verification	1-3
Performance and Resource Utilization	1-3
Installation and Licensing	1-5
OpenCore Plus Evaluation	1-5
OpenCore Plus Time-Out Behavior	1-6

Chapter 2. Getting Started

Design Flows	2-1
DSP Builder Flow	2-1
MegaWizard Plug-In Manager Flow	2-2
Generated Files	2-6
Simulate the Design	2-7
Simulating in Third-Party Simulation Tools Using NativeLink	2-7
Compile the Design and Program a Device	2-8

Chapter 3. Parameter Settings

Parameter Setting Examples	3-1
Parameter Descriptions	3-4

Chapter 4. Functional Description

Cascaded Integrator Comb Filters	4-1
Typical Frequency Response	4-2
Data Storage	4-3
Output Options	4-3
Output Data Width	4-3
Output Rounding	4-4
Hogenauer Pruning	4-4
Multi-Channel Support	4-4
Multiple Input Single Output (MISO)	4-5
Single Input Multiple Output (SIMO)	4-6
Variable Rate Change Factors for Decimation and Interpolation	4-7
FIR Filter Compensation Coefficients	4-7
Avalon Streaming Interface	4-9
Avalon-ST Interface Data Transfer Timing	4-11
Packet Data Transfers	4-11
Signals	4-12
Referenced Documents	4-13

Additional Information

Revision History	Info-1
How to Contact Altera	Info-1
Typographic Conventions	Info-2

This document describes the Altera® CIC MegaCore® function. The Altera CIC MegaCore function implements a cascaded integrator-comb filter with data ports that are compatible with the Avalon® Streaming (Avalon-ST) interface. CIC filters (also known as Hogenauer filters) are computationally efficient for extracting baseband signals from narrow-band sources using decimation, and for constructing narrow-band signals from processed baseband signals using interpolation.

CIC filters use only adders and registers, and require no multipliers to handle large rate changes. Therefore, CIC is a suitable and economical filter architecture for hardware implementation, and is widely used in sample rate conversion designs such as digital down converters (DDC) and digital up converters (DUC).

For a more detailed description, refer to [“Cascaded Integrator Comb Filters” on page 4–1](#).

Features

The Altera CIC MegaCore function supports the following features:

- Support for interpolation and decimation filters with variable rate change factors (2 to 32,000), a configurable number of stages (1 to 12), and two differential delay options (1 or 2).
- Single clock domain with selectable number of interfaces and a maximum of 1,024 channels.
- Selectable data storage options with an option to use pipelined integrators.
- Configurable input data width (1 to 32 bits) and output data width (1 to full resolution data width).
- Selectable output rounding modes (truncation, convergent rounding, rounding up, or saturation) and Hogenauer pruning support.
- Optimization for speed by specifying the number of pipeline stages used by each integrator.
- Compensation filter coefficients generation.
- Easy-to-use MegaWizard™ interface for parameterization and hardware generation.
- IP functional simulation models for use in Altera-supported VHDL and Verilog HDL simulators.
- DSP Builder ready.

Release Information

Table 1–1 provides information about this release of the Altera CIC MegaCore function.

Table 1–1. CIC MegaCore Function Release Information

Item	Description
Version	11.0
Release Date	May 2011
Ordering Code	IP-CIC
Product ID(s)	00BB
Vendor ID(s)	6AF7



For more information about this release, refer to the [MegaCore IP Library Release Notes and Errata](#).

Altera verifies that the current version of the Quartus® II software compiles the previous version of each MegaCore function. The [MegaCore IP Library Release Notes and Errata](#) report any exceptions to this verification. Altera does not verify compilation with MegaCore function versions older than one release.

Device Family Support

Table 1–2 defines the device support levels for Altera IP cores.

Table 1–2. Altera IP Core Device Support Levels

FPGA Device Families	HardCopy Device Families
Preliminary support —The IP core is verified with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. It can be used in production designs with caution.	HardCopy Companion —The IP core is verified with preliminary timing models for the HardCopy companion device. The IP core meets all functional requirements, but might still be undergoing timing analysis for the HardCopy device family. It can be used in production designs with caution.
Final support —The IP core is verified with final timing models for this device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.	HardCopy Compilation —The IP core is verified with final timing models for the HardCopy device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.

Table 1–3 shows the level of support offered by the CIC MegaCore function to each Altera device family.

Table 1–3. Device Family Support (Part 1 of 2)

Device Family	Support
Arria® GX	Final
Arria II GX	Final
Arria II GZ	Final
Cyclone®	Final

Table 1-3. Device Family Support (Part 2 of 2)

Device Family	Support
Cyclone II	Final
Cyclone III	Final
Cyclone III LS	Final
Cyclone IV GX	Final
HardCopy® II	HardCopy Compilation
HardCopy III	HardCopy Compilation
HardCopy IV E	HardCopy Compilation
HardCopy IV GX	HardCopy Compilation
Stratix®	Final
Stratix II	Final
Stratix II GX	Final
Stratix III	Final
Stratix IV GT	Final
Stratix IV GX/E	Final
Stratix V	Preliminary
Stratix GX	Final
Other device families	No support

MegaCore Verification

Before releasing a version of the CIC MegaCore function, Altera runs comprehensive regression tests to verify its quality and correctness.

Custom variations of the CIC MegaCore function are generated to exercise its various parameter options, and the resulting simulation models are thoroughly simulated with the results verified against master simulation models.

Performance and Resource Utilization

This section shows typical expected performance for a CIC MegaCore function using the Quartus II software with Cyclone III and Stratix IV devices.

The following general settings apply to all parameterizations:

- **Number of stages:** 5
- **Rate change factor:** 8
- **Differential delay:** 1
- **Integrator data storage:** Memory (whenever possible)
- **Differentiator data storage:** Memory (whenever possible)
- **Input data width:** 16
- **Output data width:** 16 (unless specified otherwise in table)
- **Output rounding:** Truncation



Cyclone III devices use combinational look-up tables (LUTs) and logic registers; Stratix IV devices use combinational adaptive look-up tables (ALUTs) and logic registers.

Table 1-4 shows the performance figures for the CIC MegaCore function, with a target f_{MAX} set to 1GHz.

Table 1-4. CIC MegaCore Function Performance

Device Family	Combinational LUTs/ALUTs	Logic Registers	Memory (3)			f _{MAX} (MHz)
			Bits	M9K	ALUTs	
Decimator, Full output resolution (Output data width = 31),1 channel per interface, 1 interface						
Cyclone III (1)	480	806	—	—	—	288
Stratix IV (2)	391	725	128	—	16	502
Decimator, Hogenauer pruning on, output data width = 16, 1 channel per interface, 1 interface						
Cyclone III (1)	388	606	—	—	—	312
Stratix IV (2)	314	525	128	—	16	526
Interpolator, Full output resolution (Output data width = 29), 1 channel per interface, 1 interface						
Cyclone III (1)	401	582	—	—	—	304
Stratix IV (2)	297	501	128	—	16	509
Interpolator, Convergent rounding, 1 channel per interface, 1 interface						
Cyclone III (1)	408	560	—	—	—	292
Stratix IV (2)	315	479	128	—	16	470
Decimator, Hogenauer pruning on, 5 channels per interface, 1 interface						
Cyclone III (1)	763	893	875	11	—	268
Stratix IV (2)	537	952	1,003	8	74	492
Decimator, Hogenauer pruning on, 5 channels per interface, 3 interfaces						
Cyclone III (1)	1,357	1,411	3,211	21	—	261
Stratix IV (2)	890	1,497	3,211	24	12	459
Decimator, Hogenauer pruning off, Truncation, 1 channel per interface, 1 interface						
Cyclone III (1)	465	746	—	—	—	302
Stratix IV (2)	391	665	128	—	16	509
Interpolator, Truncation, 5 channels per interface, 1 interface						
Cyclone III (1)	738	847	660	10	—	283
Stratix IV (2)	512	938	788	6	90	486
Interpolator, Truncation, 5 channels per interface, 3 interfaces						
Cyclone III (1)	1,501	1,657	2,426	17	—	188
Stratix IV (2)	913	1,753	2,426	20	16	345

Notes to Table 1-4:

- (1) Using EP3C10F256C6 devices.
- (2) Using EP4SGX70DF29C2X devices.
- (3) It may be possible to significantly reduce memory utilization by setting a lower target f_{MAX} .

Installation and Licensing

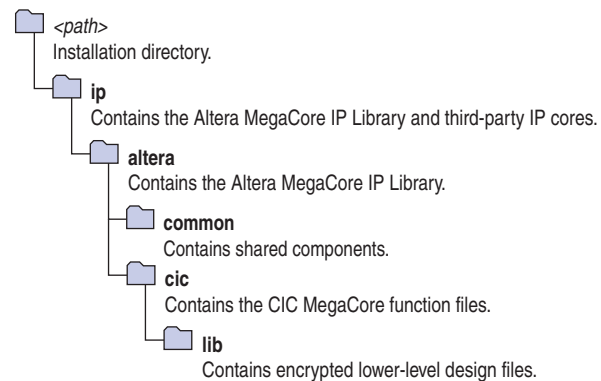
The CIC MegaCore function is part of the MegaCore IP Library, which is distributed with the Quartus® II software and can be downloaded from the Altera® website, www.altera.com.



For system requirements and installation instructions, refer to the *Altera Software Installation and Licensing* manual.

Figure 1-1 shows the directory structure after you install the CIC MegaCore function, where *<path>* is the installation directory for the Quartus II software.

Figure 1-1. Directory Structure



The default installation directory on Windows is `c:\altera\<version>`; or on Linux is `/opt/altera<version>`.

OpenCore Plus Evaluation

With Altera's free OpenCore Plus evaluation feature, you can perform the following actions:

- Simulate the behavior of a megafunction (Altera MegaCore function or AMPPSM megafunction) within your system.
- Verify the functionality of your design, as well as evaluate its size and speed quickly and easily.
- Generate time-limited device programming files for designs that include megafunctions.
- Program a device and verify your design in hardware.

You only need to purchase a license for the CIC MegaCore function when you are completely satisfied with its functionality and performance, and want to take your design to production.

After you purchase a license, you can request a license file from the Altera website at www.altera.com/licensing and install it on your computer. When you request a license file, Altera emails you a **license.dat** file. If you do not have Internet access, contact your local Altera representative.



For more information about OpenCore Plus hardware evaluation, refer to [AN320: OpenCore Plus Evaluation of Megafunctions](#).

OpenCore Plus Time-Out Behavior

OpenCore Plus hardware evaluation supports the following operation modes:

- *Untethered*—the design runs for a limited time.
- *Tethered*—requires a connection between your board and the host computer. If tethered mode is supported by all megafunctions in a design, the device can operate for a longer time or indefinitely.

All megafunctions in a device time-out simultaneously when the most restrictive evaluation time is reached. If there is more than one megafunction in a design, a specific megafunction's time-out behavior might be masked by the time-out behavior of the other megafunctions.

The untethered time-out for the CIC MegaCore function is one hour; the tethered time-out value is indefinite.

The data output signal is forced to zero when the hardware evaluation time expires.

Design Flows

The CIC MegaCore® function supports the following design flows:

- **DSP Builder:** Use this flow if you want to create a DSP Builder model that includes a CIC MegaCore function variation.
- **MegaWizard™ Plug-In Manager:** Use this flow if you would like to create a CIC MegaCore function variation that you can instantiate manually in your design.

This chapter describes how you can use a CIC MegaCore function in either of these flows. The parameterization is the same in each flow and is described in [Chapter 3, Parameter Settings](#).

After parameterizing and simulating a design in either of these flows, you can compile the completed design in the Quartus II software.

DSP Builder Flow

Altera's DSP Builder product shortens digital signal processing (DSP) design cycles by helping you create the hardware representation of a DSP design in an algorithm-friendly development environment.

DSP Builder integrates the algorithm development, simulation, and verification capabilities of The MathWorks MATLAB® and Simulink® system-level design tools with Altera Quartus® II software and third-party synthesis and simulation tools. You can combine existing Simulink blocks with Altera DSP Builder blocks and MegaCore function variation blocks to verify system level specifications and perform simulation.

In DSP Builder, a Simulink symbol for the MegaCore function appears in the MegaCore Functions library of the Altera DSP Builder Blockset in the Simulink library browser.

You can use the CIC MegaCore function in the MATLAB/Simulink environment by performing the following steps:

1. Create a new Simulink model.
2. Select the `cic_<version>` block from the MegaCore Functions library in the Simulink Library Browser, add it to your model, and give the block a unique name.
3. Double-click the `cic_<version>` block in your model to display the MegaWizard interface and parameterize the MegaCore function variation. For an example of setting parameters for the CIC MegaCore function, refer to [Chapter 3, Parameter Settings](#).
4. Click **Finish** in the MegaWizard interface to complete the parameterization and generate your CIC MegaCore function variation. For information about the generated files, refer to [Table 2–1 on page 2–6](#).
5. Connect your CIC MegaCore function variation to the other blocks in your model.

6. Simulate the CIC MegaCore function variation in your DSP Builder model.



For more information about the DSP Builder flow, refer to the *Using MegaCore Functions* chapter in the *DSP Builder User Guide*.



When you are using the DSP Builder flow, device selection, simulation, Quartus II compilation and device programming are all controlled within the DSP Builder environment.

DSP Builder supports integration with SOPC Builder using Avalon® Memory-Mapped (Avalon-MM) master/slave and Avalon Streaming (Avalon-ST) source/sink interfaces.



For more information about these interface types, refer to the *Avalon Interface Specifications*.

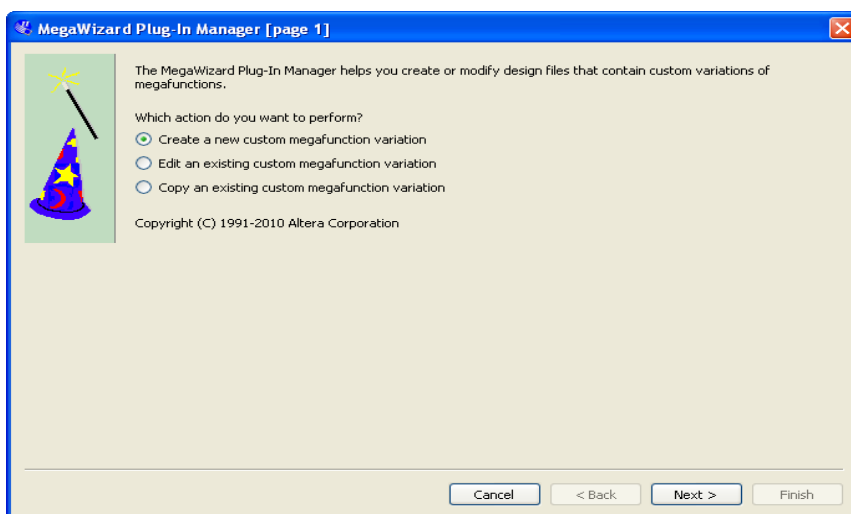
MegaWizard Plug-In Manager Flow

The MegaWizard™ Plug-in Manager flow allows you to customize a CIC MegaCore function, and manually integrate the MegaCore function variation into a Quartus II design.

Follow the steps below to use the MegaWizard Plug-in Manager flow.

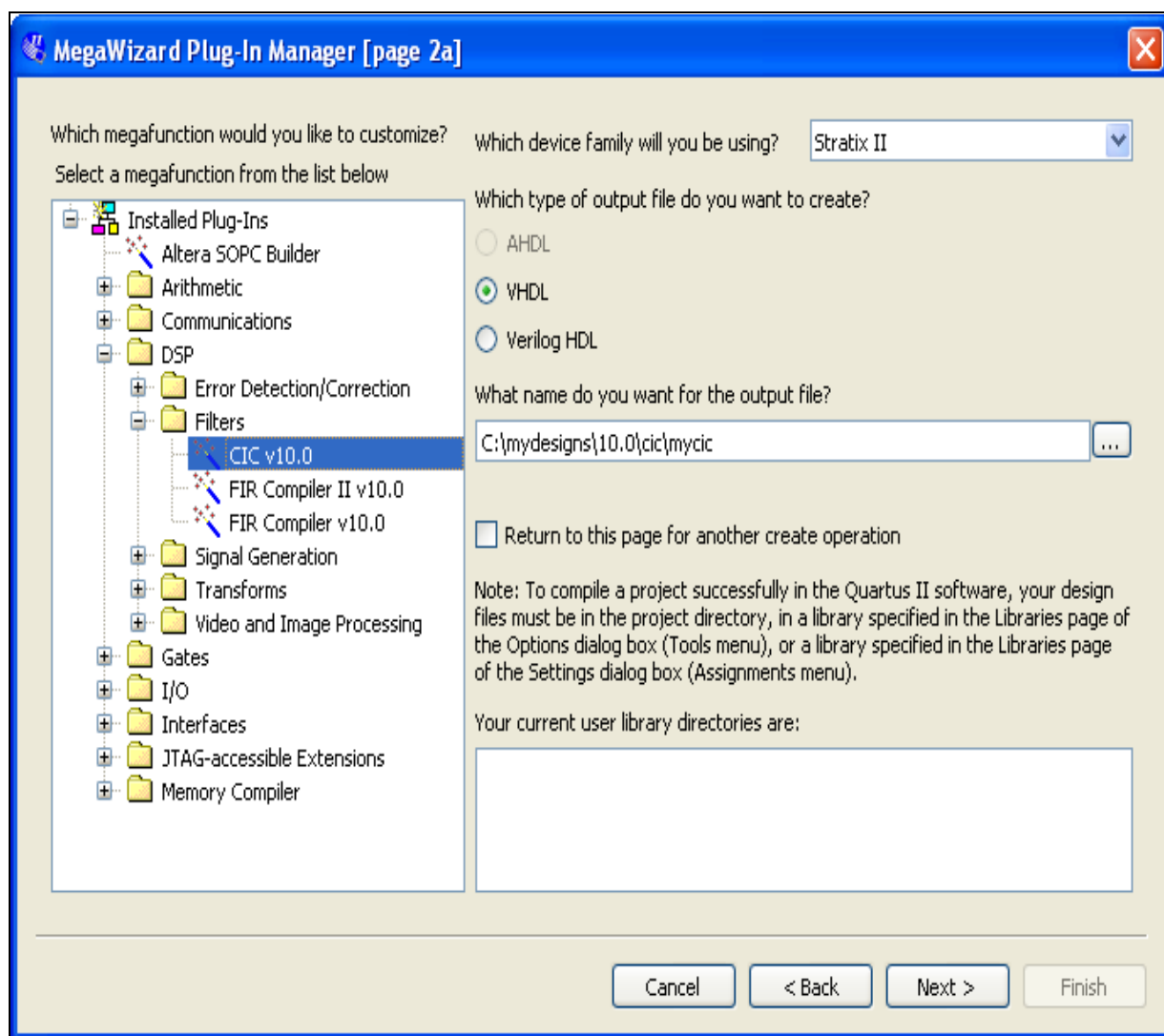
1. Create a new project using the **New Project Wizard** available from the File menu in the Quartus II software.
2. Launch **MegaWizard Plug-in Manager** from the Tools menu, and select the option to create a new custom megafunction variation ([Figure 2-1](#)).

Figure 2-1. MegaWizard Plug-In Manager



3. Click **Next** and select **CIC <version>** from the **DSP>Filters** section in the **Installed Plug-Ins** tab. ([Figure 2-2 on page 2-3](#)).

Figure 2-2. Selecting the Megafunction



4. Verify that the device family is the same as you specified in the **New Project Wizard**.
5. Select the top-level output file type for your design; the wizard supports VHDL and Verilog HDL.
6. Specify the top level output file name for your MegaCore function variation and click **Next** to display the MegaWizard interface **Parameter Settings** page. Use the MegaWizard interface to specify the required parameters for the MegaCore function variation. For an example of setting parameters for the CIC MegaCore function, refer to [Chapter 3, Parameter Settings](#).
7. Click **Next** to complete the parameterization and display the **EDA** page ([Figure 2-3 on page 2-4](#)).

Figure 2-3. EDA Page



8. On the **EDA** page, turn on **Generate Simulation Model**.



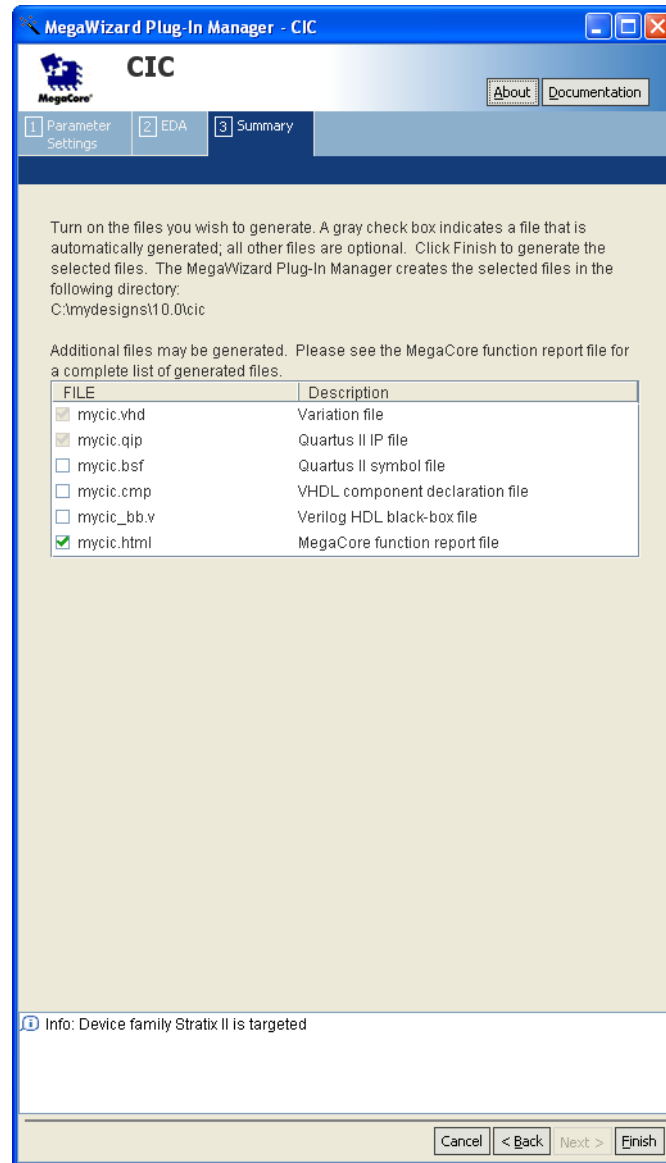
An IP functional simulation model is a cycle-accurate VHDL or Verilog HDL model produced by the Quartus II software.



Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a non-functional design.

9. Some third-party synthesis tools can use a netlist that contains only the structure of the MegaCore function, but not detailed logic, to optimize performance of the design that contains the MegaCore function. If your synthesis tool supports this feature, turn on **Generate netlist**.
10. Click Next to display the **Summary** page (Figure 2-4).

Figure 2-4. Summary Page



11. On the **Summary** tab, turn on the check boxes for the files you want to generate. A grey checkmark indicates a file that is automatically generated. All other files are optional.
12. Click **Finish** to generate the MegaCore function and supporting files. The generation phase may take several minutes to complete. The generation progress and status is displayed in a report window.

13. Click **Exit** to close the progress report window. Then click **Yes** on the **Quartus II IP Files** prompt to add the **.qip** file describing your custom MegaCore function variation to the current Quartus II project.



Refer to the Quartus II Help for more information about the MegaWizard Plug-In Manager.

Generated Files

Table 2–1 describes the generated files and other files that may be in your project directory.

The names and types of files vary depending on the variation name and HDL type you specify during parameterization. For example, a different set of files are created based on whether you create your design in Verilog HDL or VHDL.

Table 2–1. *Generated Files* (Note 1) & (Note 2)

Filename	Description
<entity name>.ocp	Encrypted OpenCore Plus file.
<entity name>.vhd or .v	A VHDL or Verilog HDL file that defines the design entity.
<variation name>.bsf	Quartus II block symbol file for the MegaCore function variation. You can use this file in the Quartus II block diagram editor.
<variation name>.cmp	A VHDL component declaration file for the MegaCore function variation. Add the contents of this file to any VHDL architecture that instantiates the MegaCore function. (3)
<variation name>.html	MegaCore function report file in hypertext markup language format which contains lists of the generated files and ports for the MegaCore function variation. (3)
<variation name>.qip	A single Quartus II IP file is generated that contains all of the assignments and other information required to process your MegaCore function variation in the Quartus II compiler. You are prompted to add this file to the current Quartus II project when you exit from the MegaWizard.
<variation name>.log	Log file.
<variation name>.vhd or .v	A VHDL or Verilog HDL file that defines a VHDL or Verilog HDL top-level description of the custom MegaCore function variation. Instantiate the entity defined by this file inside of your design. Include this file when compiling your design in the Quartus II software.
<variation name>.vho or .vo	A VHDL or Verilog HDL output file that defines the IP functional simulation model.
<variation name>_bb.v	Verilog HDL black-box file for the MegaCore function variation. Use this file when using a third-party EDA tool to synthesize your design. (3)
<variation name>_fir_comp_coeff.m	A MATLAB script for generating compensation FIR filter coefficients.
<variation name>_syn.vhd or _syn.v	A timing and resource estimation netlist for use in some third-party synthesis tools. (4)
<variation name>_nativelink.tcl	A Tcl script that can be used to assign NativeLink simulation testbench settings to the Quartus II project.
<variation name>_quartus.tcl	A Tcl script that can be used to run compilation in the Quartus II software.
<variation name>_tb_input.txt	A text file containing input data for the testbench.

Table 2-1. *Generated Files* (Note 1) & (Note 2)

Filename	Description
<variation name>_tb.vhd, or .v	A VHDL or Verilog HDL testbench file for the CIC MegaCore function variation. The VHDL file is generated when a VHDL top level has been chosen or the Verilog HDL file when a Verilog HDL top level has been chosen.

Notes to Table 2-1:

- (1) The <variation name> prefix is added automatically using the base output file name you specified in the MegaWizard Plug-In Manager.
- (2) The <entity name> prefix is added automatically. The VHDL code for each MegaCore instance is generated dynamically when you click **Finish** so that the <entity name> is different for every instance. It is generated from the <variation name> by appending **_cic**.
- (3) The **.cmp**, **_bb.v** and **.html** files are only generated when enabled in the **Summary** page of the MegaWizard interface.
- (4) The **_syn.vhd** or **_syn.v** file is only generated when enabled in the **EDA** page of the MegaWizard interface.



A MegaCore function report file containing a list of the design files and ports defined for your MegaCore function variation is saved as a HTML file if you turn on the **MegaCore function report file** check box in the MegaWizard **Summary** page.

For a full description of the signals supported on external ports for your MegaCore function variation, refer to [Table 4-3 on page 4-12](#).

Simulate the Design

You can simulate your design using the MegaWizard-generated VHDL or Verilog HDL IP functional simulation models and testbench.

The IP functional simulation model is either a **.vo** or **.vho** file, depending on the output language you specified. Compile the **.vo** or **.vho** file in your simulation environment to perform functional simulation of your custom variation of the MegaCore function.



For more information about IP functional simulation models, refer to the *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.

Simulating in Third-Party Simulation Tools Using NativeLink

You can perform a simulation in a third-party simulation tool from within the Quartus II software, using NativeLink.

The Tcl script file <variation name>_nativelink.tcl can be used to assign default NativeLink testbench settings to the Quartus II project.

To perform a simulation in the Quartus II software using NativeLink, perform the following steps:

1. Create a custom MegaCore function variation as described earlier in this chapter but ensure you specify your variation name to match the Quartus II project name.
2. Verify that the absolute path to your third-party EDA tool is set in the **Options** page under the Tools menu in the Quartus II software.
3. On the Processing menu, point to **Start** and click **Start Analysis & Elaboration**.
4. On the Tools menu, click **Tcl scripts**. In the **Tcl Scripts** dialog box, select the <variation name>_nativelink.tcl Tcl script and click **Run**. Check for a message confirming that the Tcl script was successfully loaded.

5. On the Assignments menu, click **Settings**, expand **EDA Tool Settings**, and select **Simulation**. Select a simulator under **Tool name** then in **NativeLink Settings**, select **Compile test bench** and click **Test Benches**.
6. On the Tools menu, point to **EDA Simulation Tool** and click **Run EDA RTL Simulation**.

The Quartus II software selects the simulator, and compiles the Altera libraries, design files, and testbenches. The testbench runs and the waveform window shows the design signals for analysis.



For more information, refer to the *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.

Compile the Design and Program a Device

You can use the Quartus II software to compile your design. After a successful compilation, you can program the targeted Altera device and verify the design in hardware.



For instructions on compiling and programming your design, and more information about the MegaWizard Plug-In Manager flow, refer to the Quartus II Help.

This chapter gives an example of how to parameterize a CIC MegaCore function.

The **Parameter Settings** page provides the same options whether the MegaWizard interface has been opened from the DSP Builder or MegaWizard Plug-In Manager flow. However, the **EDA** and **Summary** tabs are not visible when you are using the DSP Builder flow.

For information about opening the MegaWizard interface, refer to the “[Design Flows](#)” on [page 2-1](#).



The MegaWizard interface only allows you to select legal combinations of parameters, and warns you of any invalid configurations.

Parameter Setting Examples

To parameterize your CIC MegaCore function, follow these steps:

1. Specify your required values in the **Parameter Settings: Architecture** page. For example [Figure 3-1 on page 3-2](#) shows the parameter setting listed in [Table 3-1](#).

Table 3-1. Example Parameters for the CIC MegaCore Function Architecture Tab

Parameter	Value
Filter type	Decimator
Number of stages	3
Differential delay	1
Rate change factor	4
Enable variable rate change factor	Off
Number of interfaces	1
Number of channels per interface	7
Integrator data storage	Memory
RAM type of integrator data storage	AUTO
Differentiator data storage	Memory
RAM type of differentiator data storage	AUTO
Use pipelined integrators	On
Number of pipeline stages per integrator	2



If you enable the variable rate change factor option you can also specify minimum and maximum values instead of specifying an actual rate change factor. The data storage options are only available when the number of channels per interface is greater than 4.

For more information about these parameters, refer to [Table 3-3 on page 3-4](#).

Figure 3-1. Architecture Page

MegaWizard Plug-In Manager - CIC

CIC

About Documentation

1 Parameter Settings 2 EDA 3 Summary

Architecture > Input/Output Options >

Device Family

Target: Stratix II

Filter Specifications

Filter type: Decimator

Number of stages: 3

Differential delay: 1

Rate change factor: 4

Variable Rate Change Factor Options

☐ Enable variable rate change factor

Minimum: 2 Maximum: 3

Multi-channel Options

Number of interfaces: 1

Number of channels per interface: 7

Data Storage Options

Integrator data storage: Memory

RAM type of integrator data storage: AUTO

Differentiator data storage: Memory

RAM type of differentiator data storage: AUTO


Optimize for Speed

☒ Use pipelined integrators

Number of pipeline stages per integrator: 1

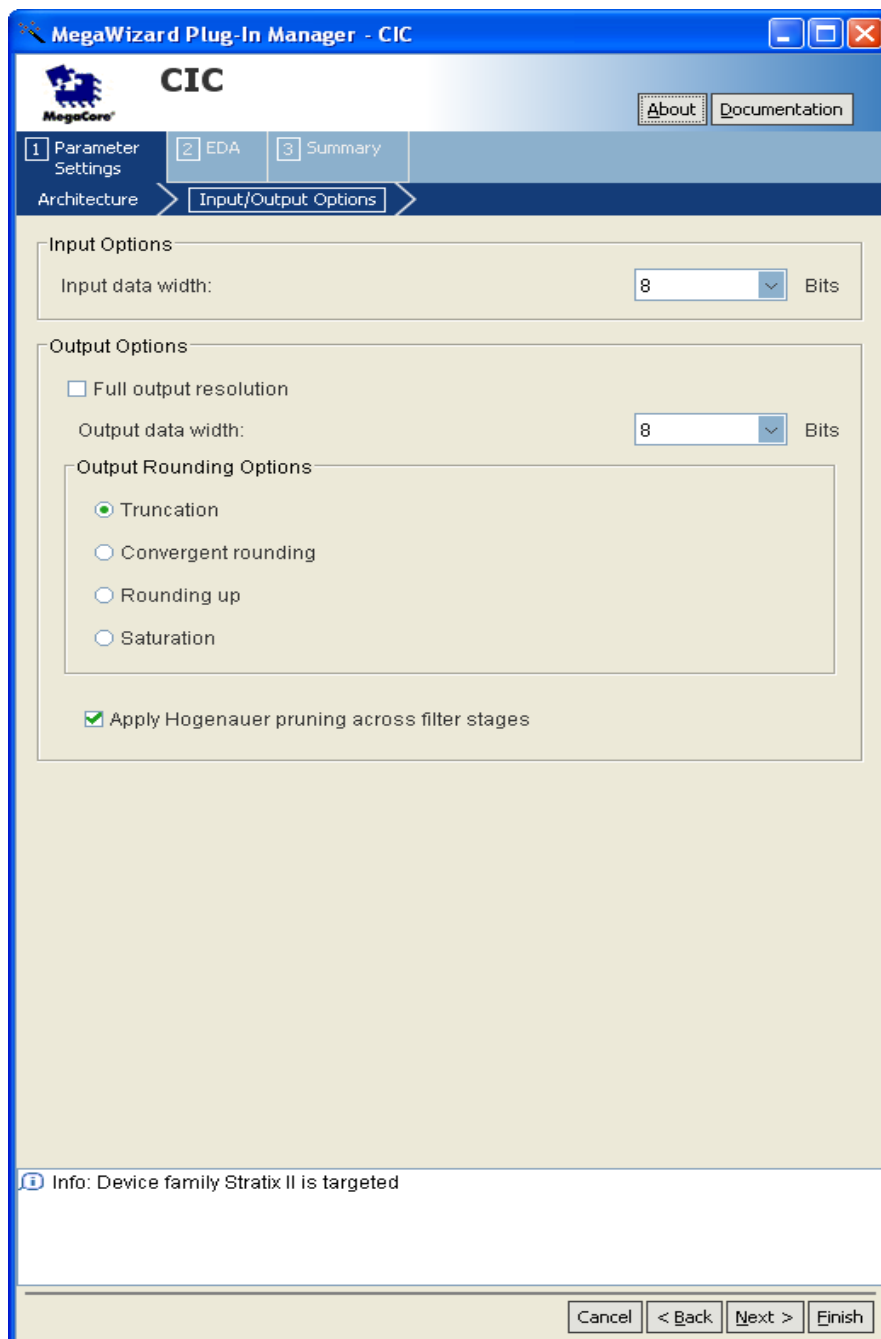
Info: Device family Stratix II is targeted

Cancel < Back Next > Finish

 The **Parameter Settings - Architecture** page displays the target device family as a read-only field showing the current selection. The target device can be specified in Signal Compiler if you are using the DSP Builder flow, or the **New Project Wizard** if you are using the MegaWizard Plug-In Manager flow.

2. Click **Next** to display the **Parameter Settings: Input/Output Options** page (Figure 3-2).

Figure 3-2. Input/Output Options Page



MegaWizard Plug-In Manager - CIC

CIC

About Documentation

1 Parameter Settings 2 EDA 3 Summary

Architecture > Input/Output Options >

Input Options

Input data width: 8 Bits

Output Options

☐ Full output resolution

Output data width: 8 Bits

Output Rounding Options

☒ Truncation

☐ Convergent rounding

☐ Rounding up

☐ Saturation

☒ Apply Hogenauer pruning across filter stages

Info: Device family Stratix II is targeted

Cancel < Back Next > Finish

3. Use the **Parameter Settings: Input/Output Options** page to specify the values listed in [Table 3-2](#).

Table 3-2. Example Parameters for the CIC MegaCore Function Input/Output Options Tab

Parameter	Value
Input data width	8
Full output resolution	Off
Output data width	8
Output rounding options	Truncation
Apply Hogenauer pruning across filter stages	On

For more information about these parameters, refer to [Table 3-4 on page 3-5](#).

Parameter Descriptions

This section describes the CIC MegaCore function parameters, which can be set in the MegaWizard interface (Refer to [“Parameter Setting Examples” on page 3-1](#)).

[Table 3-3](#) shows the parameters that can be set in the **Architecture** page.

Table 3-3. CIC MegaCore Function Architecture Page (Part 1 of 2)

Parameter	Value	Description
Target	Refer to Table 1-3 on page 1-2 for the list of supported devices	Displays the target device family that you specified when you created the Quartus II project.
Filter type	Decimator, Interpolator	You can select whether to implement a decimator or interpolator.
Number of stages	1–12	Specifies the required number of stages.
Differential delay	1, 2	Specifies the differential delay in cycles.
Rate change factor	2–32000	Specifies the rate change factor.
Enable variable rate change factor	On or Off	Turn on to enable a variable rate change factor that can be changed at runtime. When this option is on, the Rate change factor parameter is not available but you can specify minimum and maximum values.
Number of interfaces	1–128	Specifies the number of MISO inputs or SIMO outputs. (1)
Number of channels per interface	1–1024	Specifies the number of channels per interface. (1)
Integrator data storage	Logic Element, Memory	You can select whether to implement the integrator data storage as logic elements or memory. (2)
RAM type of integrator data storage	AUTO, M512, M4K, M9K, M144K, MLAB	When Memory is selected, you can select the RAM type used for integrator data storage. (4)
Differentiator data storage	Logic Element, Memory	You can select whether to implement the differentiator data storage as logic elements or memory. (3)
RAM type of differentiator data storage	AUTO, M512, M4K, M9K, M144K, MLAB	When Memory is selected, you can select the RAM type used for differentiator data storage. (4)

Table 3-3. CIC MegaCore Function Architecture Page (Part 2 of 2)

Parameter	Value	Description
Use pipelined integrators	On or Off	Turn on to use pipelined integrators. This option is available when the Number of channels per interface is greater than or equal to 2 (or greater than or equal to 6, when the Memory option is selected for integrator data storage). Use this option for multichannel designs that have large input bit width and require high f_{MAX} . This option is recommended for designs targeting Stratix family devices, but not for Cyclone family devices.
Number of pipeline stages per integrator	1–4	Specifies the number of pipeline stages used by each integrator. Adding additional integrators can improve f_{MAX} but increases the resource utilization. The maximum number of pipeline stages, that can be used, is dependent on the number of channels and whether Memory or Logic Cells is selected for integrator data storage. When Memory is selected, the maximum number of pipeline stages equals the number of channels minus 5. When Logic Cells is selected, the maximum number of pipeline stages equals the number of channels.

Notes to Table 3-3:

- (1) The product of the **Number of interfaces** and the **Number of channels per interface** must be no more than 1024.
- (2) The **Memory** option is available for integrator data storage when the **Number of channels per interface** is greater than 4.
- (3) The **Memory** option is available for differentiator data storage when the product of the **Differential delay**, **Number of channels per interface** and **Number of interfaces** is greater than 4.
- (4) The options available depend on the target device family. When **AUTO** is selected, the Quartus II software automatically selects the optimum RAM type for the currently selected device family.

Table 3-4 shows the parameters that can be set in the **Input/Output Options** page.

Table 3-4. CIC MegaCore Function Input/Output Options Page Parameters

Parameter	Value	Description
Input data width	1–32	Specifies the input data width in bits.
Full output resolution	On, Off	Turn on to enable full output resolution. When selected, the output data width is set to its maximum and the output rounding options are disabled.
Output data width	1 to calculated maximum data width	Specifies the output data width in bits.
Output Rounding Options	Truncation, Convergent rounding, Rounding up, Saturation	Select the required rounding output mode. Note that the saturation limit is the maximum value for overflow or the minimum value for negative overflow. (1)
Apply Hogenauer pruning across filter stages	On, Off	This option is available only when a Decimator filter type is selected in the Architecture page. Turn on if you want to apply Hogenauer pruning across the filter stages.

Note for Table 3-4:

- (1) Refer to “**Output Rounding**” on page 4-4 for more information about these options.

Cascaded Integrator Comb Filters

Cascaded Integrator Comb (CIC) filters are widely used in modern communication systems. Using CIC filters provides a silicon efficient architecture for performing sample rate conversion. This is achieved by extracting baseband signals from narrow-band sources using decimation, and constructing narrow-band signals from processed baseband signals using interpolation. The key advantage of CIC filters is that they use only adders and registers, and do not require multipliers to implement in hardware for handling large rate changes.

A CIC filter (also known as a Hogenauer filter) can be used to perform either decimation or interpolation. A decimation CIC filter comprises a cascade of integrators (called the integrator section), followed by a down sampling block (decimator) and a cascade of differentiators (called the differentiator or comb section). Similarly an interpolation CIC filter comprises a cascade of differentiators, followed by an up sampling block (interpolator) and a cascade of integrators.

In a CIC filter, both the integrator and comb sections have the same number of integrators and differentiators. Each pairing of integrator and differentiator is called a stage. The number of stages (N) has a direct effect on the frequency response of a CIC filter. The response of the filter is determined by configuring the number of stages N , the rate change factor R and the number of delays in the differentiators (called the differential delay) M . In practice, the differential delay is set to 1 or 2.

Figure 4–1 shows an integrator.

Figure 4–1. Integrator

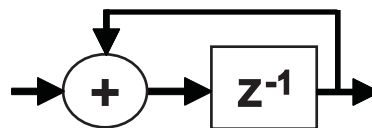


Figure 4–2 shows a differentiator.

Figure 4–2. Differentiator

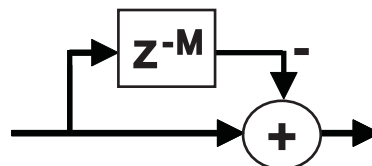


Figure 4-3 shows three integrators and three differentiators combined using a decimator to make a three stage decimation CIC filter.

Figure 4-3. Three Stage Decimation CIC Filter

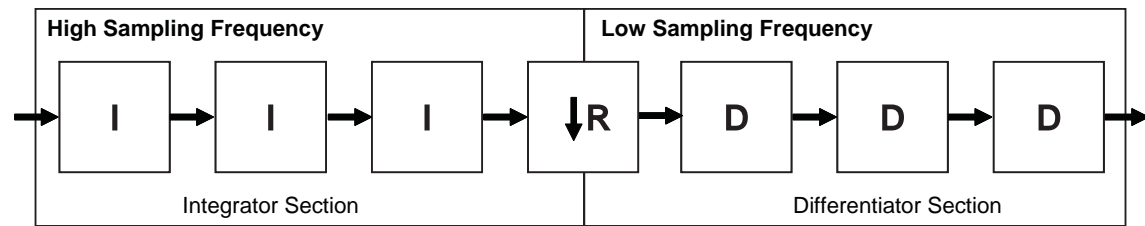
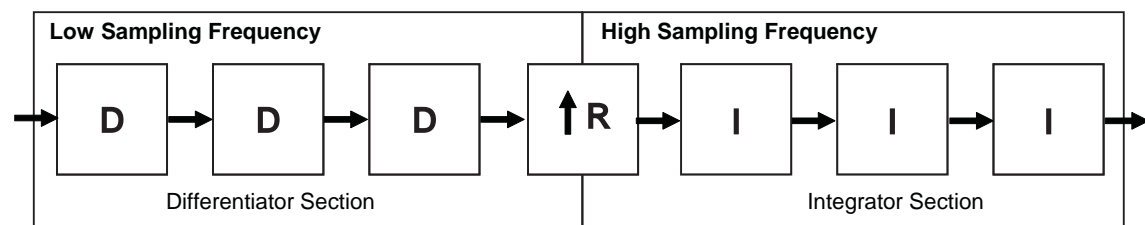


Figure 4-4 shows three differentiators and three integrators combined using an interpolator to make a three stage interpolation CIC filter.

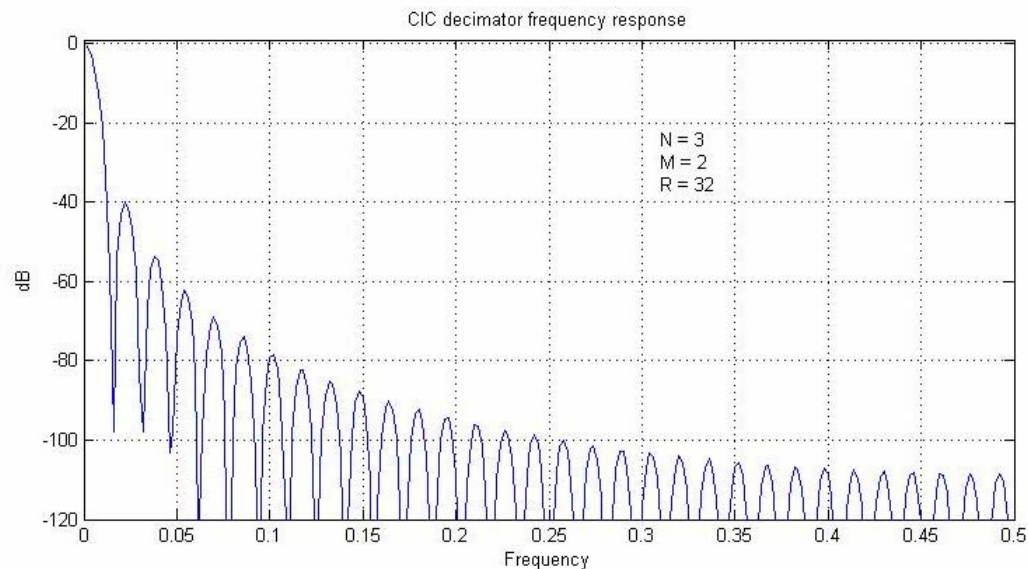
Figure 4-4. Three Stage Interpolation Filter



Typical Frequency Response

Figure 4-5 shows the frequency response for a CIC decimation filter with $N = 3$, $M = 2$ and $R = 32$.

Figure 4-5. Three stage CIC Decimation Filter Frequency Response



Data Storage

Data storage in the integrators and differentiators can utilize either logic cells, or memory blocks. The type of data storage can be selected in the MegaWizard interface when the data storage options are enabled. The memory types are then selected automatically by the Quartus II software depending on the chosen device.

For information about when the memory option is enabled, refer to [Table 3–3 on page 3–4](#).

Output Options

You can select output options for the output data bit width and rounding options.

Output Data Width

For a decimation filter, the gain at the output of the filter is:

$$G = (RM)^N$$

Therefore, the data width at the output stage for if full resolution is:

$$B_{out} = \lceil B_{in} + N \log_2(RM) \rceil$$

where B_{in} is the input data width.



A data width of B_{out} is required for each integrator and differentiator for no data loss.

If you have selected an output data width that is smaller than the full output resolution data width, the Hogenauer pruning technique can be applied to reduce the data widths across the filter stages and hence the overall resource utilization.

For an interpolation filter, the gain at each filter stage is:

$$G_i = \begin{cases} 2^i & i = 1, 2, \dots, N \\ \frac{2^{2N-1}(RM)^{i-N}}{R} & i = N+1, \dots, 2N \end{cases}$$

Hence the required data width at the i th stage is:

$$W_i = \lceil B_{in} + \log_2(G_i) \rceil$$

and the data width at the output stage is:

$$B_{out} = \lceil B_{in} + N \log_2(RM) - \log_2(R) \rceil$$

where B_{in} is the input data width.

When the differential delay is one, the bit width at each integrator stage is increased by one to ensure stability.



For more information about these calculations, refer to Hogenauer, Eugene. *An Economical Class of Digital Filters For Decimation and Interpolation*, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-29, pp. 155-162, April 1981.

Output Rounding

For high rate change factors, the maximum required data width for no data loss is large for many practical cases. To reduce the output data width to the input level, quantization is normally applied at the end of the output stage. In this case, the following rounding or saturation options are available:

- **Truncation:** The LSBs are dropped. (This is equivalent to rounding to minus infinity.)
- **Convergent rounding.** Also known as unbiased rounding. Rounds to the nearest even number. If the most significant deleted bit is one, and either the least significant of the remaining bits or at least one of the other deleted bits is one, then one is added to the remaining bits.
- **Round up:** Also known as rounding to plus infinity. Adds the MSB of the discarded bits for positive and negative numbers via the carry in.
- **Saturation:** Puts a limit value (upper limit in the case of overflow, or lower limit in the case of negative overflow) at the output when the input exceeds the allowed range. The upper limit is $+2^{n-1}$ and lower limit is -2^n .



These rounding options can only be applied to the output stage of the filter. The data widths at the intermediate stages are not changed. The next section describes cases where the data width at the intermediate stages can be changed.

Hogenauer Pruning

Hogenauer pruning [Reference] is a technique that utilizes truncation or rounding in intermediate stages with the retained number of bits decreasing monotonically from stage to stage, while the total error introduced is still no greater than the quantization error introduced by rounding the full precision output. This technique helps to reduce the number of logic cells used by the filter and gives better performance.

The existing algorithms for computing the Hogenauer bit width growth for large N and R values are computationally expensive.



For more information about these algorithms, refer to U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 2nd Edition, Springer, 2004.

The CIC MegaCore function has pre-calculated Hogenauer pruning bit widths stored within the MegaCore function. There is no need to wait for Hogenauer pruning bit widths to be calculated if Hogenauer pruning is enabled for a decimation filter.



Hogenauer pruning is only available to decimation filters when the selected output data width is smaller than the full output resolution data width.

Multi-Channel Support

There are often many channels of data in a digital signal processing (DSP) system that require filtering by CIC filters with the same configuration. These can be combined into one filter, which shares the adders that exist in each stage and reduces the overall resource utilization.

This combined filter uses fewer resources than using many individual CIC filters. For example, a two-channel parallel filter requires two clock cycles to calculate two outputs. The resulting hardware would need to run at twice the data rate of an individual filter. This is especially useful for higher rate changes where adders grow particularly large.



To minimize the number of logic elements, a multiple input single output (MISO) architecture can be used for decimation filters, and a single input multiple output (SIMO) architecture for interpolation filters as described in the following sections.

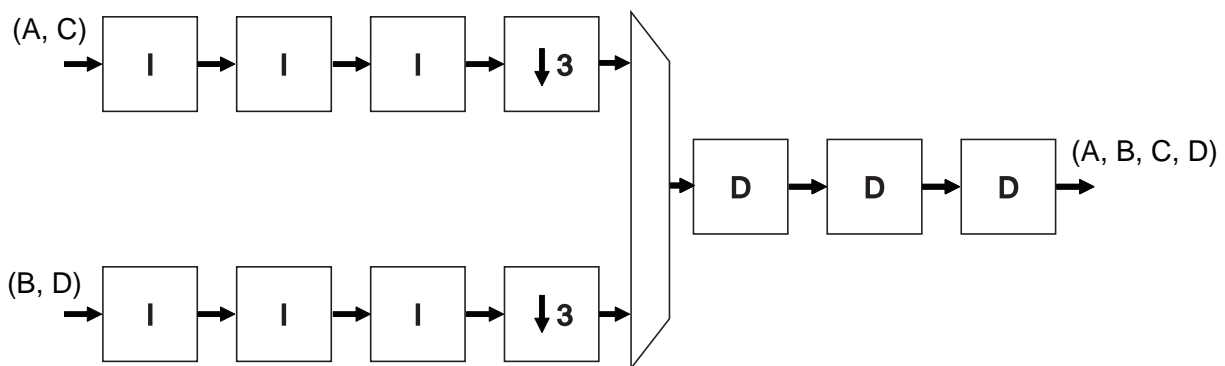
Multiple Input Single Output (MISO)

In many practical designs, channel signals come from different input interfaces. On each input interface, the same parameters including rate change factors are applied to the channel data that the CIC filter is going to process. The CIC MegaCore function allows the flexibility to exploit time sharing of the low rate differentiator sections.

This is achieved by providing multiple input interfaces and processing chains for the high rate portions, then combining all of the processing associated with the lower rate portions into a single processing chain. This strategy can lead to full utilization of the resources and represents the most efficient hardware implementation. These architectures are known as multiple input single output (MISO) decimation filters.

Figure 4-6 shows an example of the MISO architecture for a CIC filter that processes a total of four channels. In this example, the symbols *A*, *B*, *C*, *D* are multiplexed into one output *A*, *B*, *C*, *D*.

Figure 4-6. Multiple Input Single Output Architecture



The sampling frequency of the input data is such that it is only possible to time multiplex two channels per bus, therefore the CIC filter must be configured with two input interfaces. Because two interfaces are required, the rate change factor must also be at least two to exploit this architecture. Up to 1,024 channels can be supported by using multiple input interfaces in this way.



MISO architecture is applied when a decimation filter type is chosen and the number of interfaces selected in the MegaWizard interface is greater than one.

Single Input Multiple Output (SIMO)

Single input multiple output (SIMO) is a feature associated with interpolation CIC filters. In this architecture, all the channel signals presented for filtering come from a single input interface.

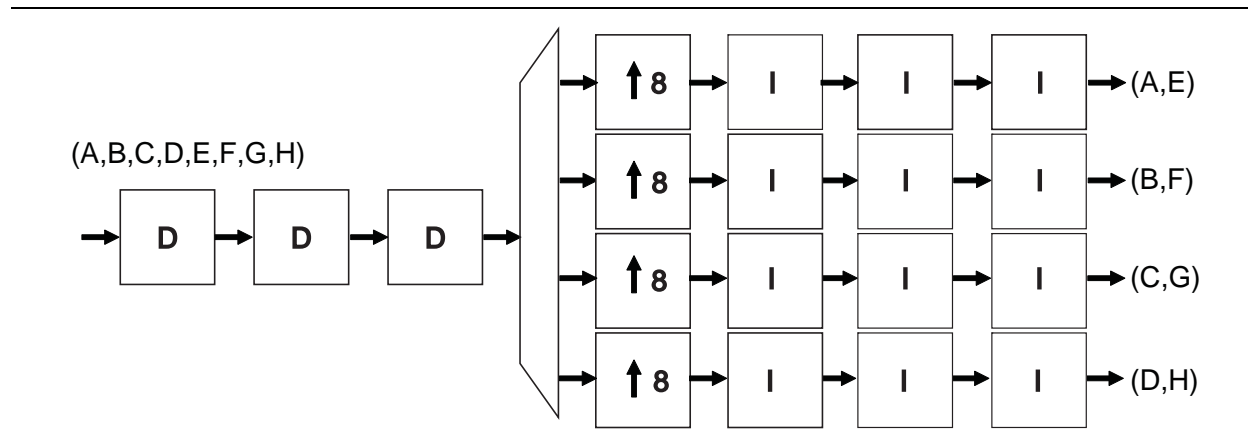
Like the MISO case, it is possible to share the low sampling rate differentiator section amongst more channels than the higher sampling frequency integrator sections. Therefore, this architecture features a single instance of the differentiator section, and multiple parallel instances of the integrator sections.

After processing by the differentiator section, the channel signals are split into multiple parallel sections for processing in a high sampling frequency by the integrator sections.

Figure 4-7 shows an example of the SIMO architecture for a CIC filter that processes a total of eight channels.

In this example, the symbols *A, B, C, D, E, F, G, H* are demultiplexed into four outputs *A, E; B, F; C, G; and D, H*.

Figure 4-7. Single Input Multiple Output Architecture



The required sampling frequency of the output data is such that it is only possible to time multiplex two channels per bus. Therefore the CIC filter must be configured with four output interfaces. Because four interfaces are required, the rate change factor must also be at least four to exploit this architecture, but in this example a rate change of eight is illustrated.



SIMO architecture is applied when an interpolation filter type is chosen and the number of interfaces selected in the MegaWizard interface is greater than one.

The total number of input channels must be a multiple of the number of interfaces. To satisfy this requirement, you may need to either insert dummy channels or use more than one CIC MegaCore function.

Data is transferred as packets using Avalon Streaming (Avalon-ST) interfaces. A general description of these interfaces is given in “[Avalon Streaming Interface](#)” on page 4-9.



For an example design using multi-channel MISO and SIMO architectures, refer to *AN442: Tool Flow Design of Digital IF for Wireless Systems*.

Variable Rate Change Factors for Decimation and Interpolation

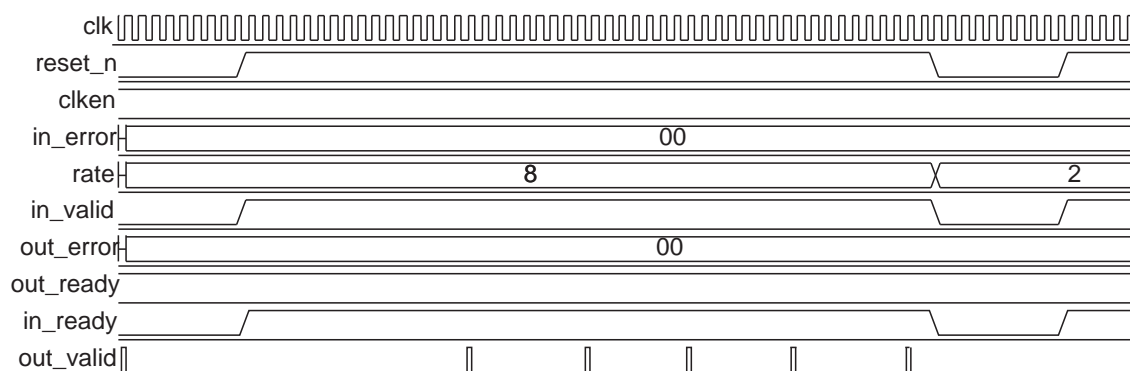
You can optionally set minimum and maximum values for the decimator or interpolator rate change factors and enable the rate change factors to be set at run time. When these options are set, an additional `rate` port is provided that can be used to specify the rate change factor. For information about how to enable the options for variable rate change factors, refer to “[Parameter Setting Examples](#)” on page 3-1.



If variable rate change factors are enabled, the MegaCore function must be reset when the rate change factor is changed otherwise the previous memory/register values would be used. The filter mode (interpolation or decimation) cannot be changed at run time.

[Figure 4-8](#) illustrates the input and output timing relationships for a variable rate change decimation CIC filter. Note how the `out_valid` signal changes its period according to the variable rate change.

Figure 4-8. Variable Rate Change Decimation CIC Filter Timing Diagram



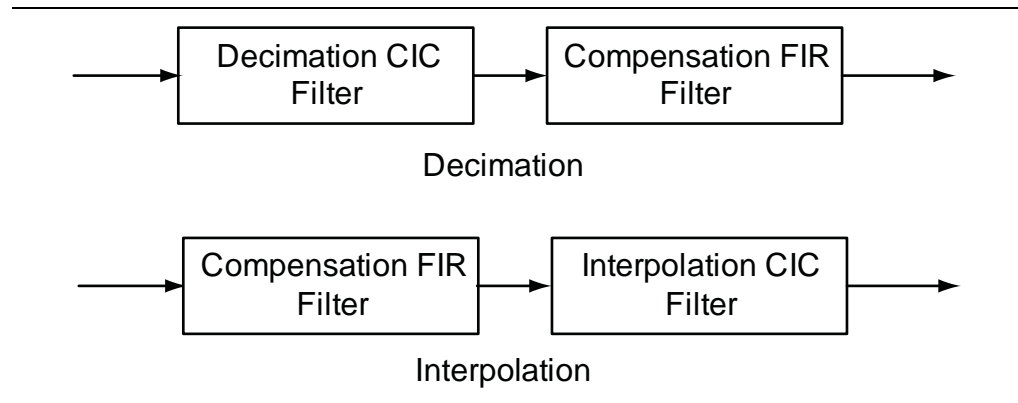
FIR Filter Compensation Coefficients

CIC filters have a low-pass filter characteristic. There are only three parameters (the rate change factor R , the number of stages N , and the differential delay M) that can be modified to alter the passband characteristics and aliasing/imaging rejection.

However, due to their drooping passband gains and wide transition regions, CIC filters alone cannot provide the flat passband and narrow transition region filter performance that is typically required in decimation or interpolation filtering applications.

This problem can be alleviated by connecting the decimation or interpolation CIC filter to a compensation FIR filter which narrows the output bandwidth and flattens the passband gain (Figure 4-9).

Figure 4-9. Using a CIC Compensation FIR Filter



You can use a frequency sampling method to determine the coefficients of a FIR filter that equalizes the undesirable passband droop of the CIC and construct an ideal frequency response.

The ideal frequency response is determined by sampling the normalized magnitude response of the CIC filter before inverting the response.

Generally, it is only necessary to equalize the response in the passband, but you can sample further than the passband to fine tune the cascaded response of the filter chain.

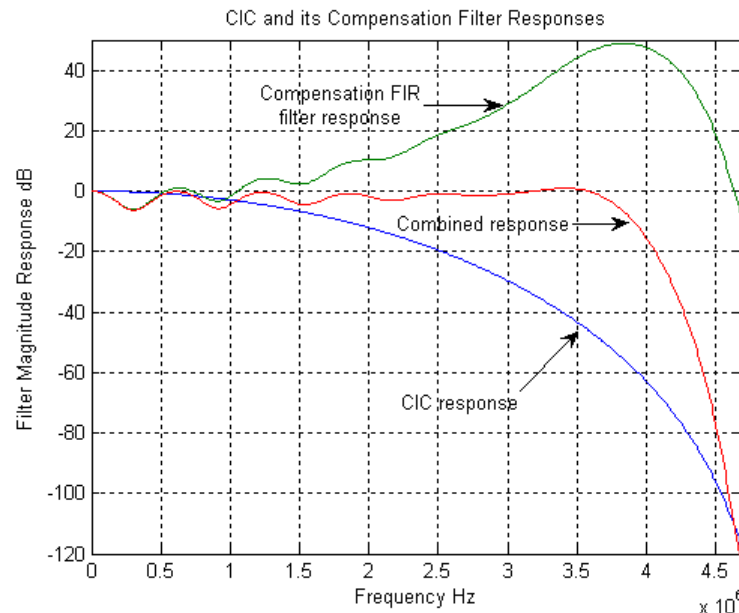
A MATLAB script `<variation_name>_fir_comp_coeff.m` is generated in the project directory by the CIC MegaCore function. You can run this script in MATLAB to generate FIR coefficients that provide appropriate passband equalization. The generated coefficients are saved in a text file, which is ready for use by the Altera FIR Compiler MegaCore function.

The MATLAB script requires the following parameters for the compensation FIR filter:

- L : FIR filter length, which is same as the number of taps or the number of coefficients
- F_s : FIR filter sample rate in Hz before decimation/interpolation
- F_c : FIR filter cutoff frequency in Hz
- B : Coefficient bit width if coefficients are written in fixed-point numbers

An example of compensation filter response is shown in Figure 4-10 on page 4-9.

Figure 4–10. CIC and Compensation Filter Responses



For more information, refer to [AN455: Understanding CIC Compensation Filters](#).

Avalon Streaming Interface

The Avalon® Streaming (Avalon-ST) interface is an evolution of the Atlantic™ interface. The Avalon-ST interface defines a standard, flexible, and modular protocol for data transfers from a source interface to a sink interface and simplifies the process of controlling the flow of data in a datapath.

Avalon-ST interface signals can describe traditional streaming interfaces supporting a single stream of data without knowledge of channels or packet boundaries. Such interfaces typically contain data, ready, and valid signals.

The Avalon-ST interface can also support more complex protocols for burst and packet transfers with packets interleaved across multiple channels.

The Avalon-ST interface inherently synchronizes multi-channel designs, which allows you to achieve efficient, time-multiplexed implementations without having to implement complex control logic.

The Avalon-ST interface supports backpressure, which is a flow control mechanism where a sink can signal to a source to stop sending data. The sink typically uses backpressure to stop the flow of data when its FIFO buffers are full or when there is congestion on its output.

When designing a datapath which includes the CIC MegaCore function, you may not need backpressure if you know that the downstream components can always receive data.

The *Avalon Interface Specifications* define parameters which can be used to specify any type of Avalon-ST interface. Table 4–1 lists the values of these parameters that are defined for the Avalon-ST interfaces used by the CIC MegaCore function. All parameters not explicitly listed in the table have undefined values.

Table 4–1. Avalon-ST Interface Parameters

Parameter Name	Value
READY_LATENCY	0
BITS_PER_SYMBOL	data width
SYMBOLS_PER_BEAT	(Note 1), (Note 2), (Note 3)
SYMBOL_TYPE	signed
ERROR_DESCRIPTION	00: No error 01: Missing startofpacket (SOP) 10: Missing endofpacket (EOP) 11: Unexpected EOP or any other error

Notes for Table 4–1:

- (1) For single input, single output architectures, there is one symbol per beat at the source and the sink.
- (2) For MISO architectures, there are <number of interfaces> symbols per beat at the sink, and a single symbol per beat at the source.
- (3) For SIMO architectures, there are <number of interfaces> symbols per beat at the source, and a single symbol per beat at the sink.

The *Avalon Interface Specifications* define many signal types many of which are optional.

Table 4–2 lists the signal types used by the Avalon-ST interfaces for the CIC MegaCore function.

Table 4–2. Avalon-ST Interface Signal Types

Signal Type	Width
ready	1
valid	1
data	data width
channel	$\log_2(\text{number of channels})$
error	2
startofpacket	1
endofpacket	1

Any signal type not explicitly listed in the table is not used by the CIC MegaCore function

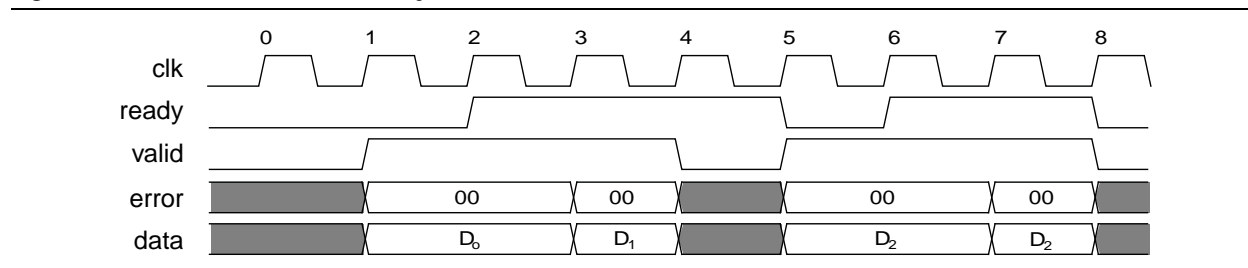


For a full description of the Avalon-ST interface protocol, refer to the *Avalon Interface Specifications*.

Avalon-ST Interface Data Transfer Timing

Figure 4-11 illustrates the data transfer timing.

Figure 4-11. Avalon-ST Interface Timing with READY_LATENCY=0



The source provides data and asserts `valid` on cycle 1, even though the sink is not ready. The source waits until cycle 2, when the sink does assert `ready`, before moving onto the next data cycle. In cycle 3, the source drives data on the same cycle and because the sink is ready to receive it, the transfer occurs immediately. In cycle 4, the sink asserts `ready`, but the source does not drive valid data.

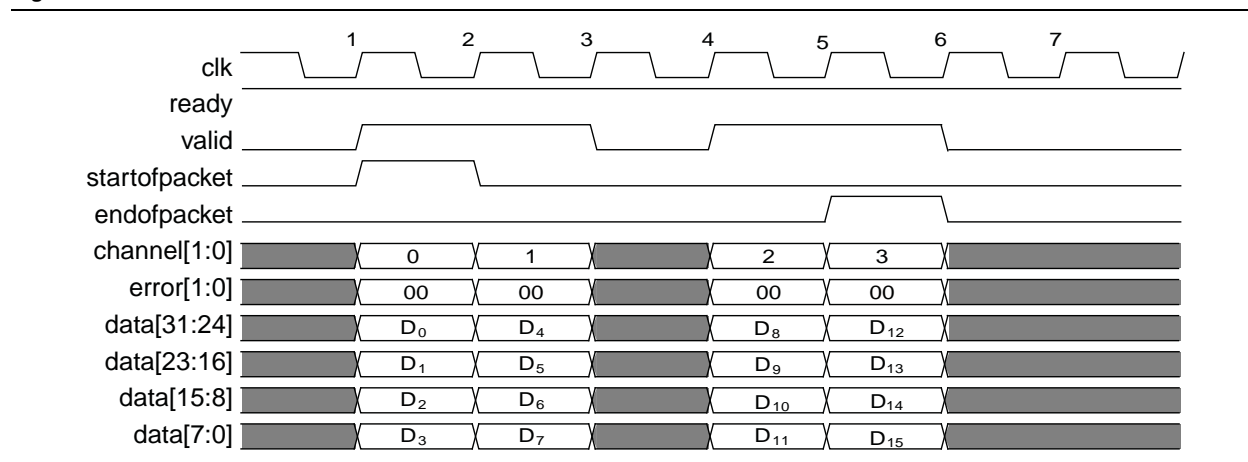
Packet Data Transfers

A beat is defined as the transfer of one unit of data between a source and sink interface. This unit of data may consist of one or more symbols and makes it possible to support modules that convey more than one piece of information about each valid cycle.

Packet data transfers are used for multichannel transfers. Two additional signals (`startofpacket` and `endofpacket`) are defined to implement the packet transfer.

Figure 4-12 shows an example where four symbols are transferred on each beat. This multiple symbols per beat scenario applies to both the sink interface on MISO CIC filters and the source interface of SIMO CIC filters. All other interfaces operate with a single symbol per beat, but the interfaces also support multiple channels using packets.

Figure 4-12. Packet Data Transfer



The data transfer in Figure 4-12 occurs on cycles 1, 2, 4, and 5, when both `ready` and `valid` are asserted.

During cycle 1, `startofpacket` is asserted, and the first four bytes of packet are transferred. During cycle 5, `endofpacket` is asserted indicating that this is the end of the packet. The `channel` signal indicates the channel index associated with the data. For example, on cycle 1, the data D_0 , D_1 , D_2 , and D_3 associated with channel 0 are available.

Signals

Table 4–3 lists the input and output signals for the CIC MegaCore Function.

Table 4–3. CIC MegaCore Function Signals (Part 1 of 2)

Signal	Direction	Description
<code>clk</code>	Input	Clock signal used to clock all internal registers.
<code>reset_n</code>	Input	Active low reset signal. The CIC MegaCore function must always be reset before receiving data. If the MegaCore function is not reset, the CIC filter may produce unexpected results due to feedback signals.
<code>clken</code>	Input	Optional top-level clock enable.
<code>in_data</code>	Input	Sample input. For multiple input cases, the input data ports are named as <code>in0_data</code> , <code>in1_data</code> , and so on.
<code>in_endofpacket</code>	Input	Marks the end of the incoming sample group. If there are N channels, the end of packet signal must be high when the sample belonging to the last channel, that is channel $N-1$, is presented at <code>in_data</code> .
<code>in_error</code>	Input	Error signal indicating Avalon Streaming protocol violations on input side: <ul style="list-style-type: none"> ■ 00: No error ■ 01: Missing start of packet ■ 10: Missing end of packet ■ 11: Unexpected end of packet Other types of error are also marked as 11.
<code>in_ready</code>	Output	Indicates when the MegaCore function is able to accept data.
<code>in_startofpacket</code>	Input	Marks the start of the incoming sample group. The start of packet is interpreted as a sample from channel 0.
<code>in_valid</code>	Input	Asserted when data at <code>in_data</code> is valid. When <code>in_valid</code> is not asserted, processing is stopped until valid is re-asserted. If <code>clken</code> is 0, <code>in_valid</code> will not be asserted.
<code>out_channel</code>	Output	Specifies the channel whose result is presented at <code>out_data</code> .
<code>out_data</code>	Output	Filter output. The data width depends on the parameter settings. For multiple output cases, the output data ports are named as <code>out0_data</code> , <code>out1_data</code> , and so on.
<code>out_endofpacket</code>	Output	Marks the end of the outgoing result group. If '1', a result corresponding to channel $N-1$ is output, where N is the number of channels.
<code>out_error</code>	Output	Error signal indicating Avalon Streaming protocol violations on source side: <ul style="list-style-type: none"> ■ 00: No error ■ 01: Missing start of packet ■ 10: Missing end of packet ■ 11: Unexpected end of packet Other types of errors may also be marked as 11.

Table 4-3. CIC MegaCore Function Signals (Part 2 of 2)

Signal	Direction	Description
out_ready	Input	Asserted by the downstream module if it is able to accept data.
out_startofpacket	Output	Marks the start of the outgoing result group. If '1', a result corresponding to channel 0 is output.
out_valid	Output	Asserted by the MegaCore function when there is valid data to output.
rate	Input	This signal is available when the variable rate change factor option is enabled, and can be used to change the decimation/interpolation rate during run time. It has the size $\text{Ceil}(\log_2(\text{maximum rate}))$.

Referenced Documents

Altera application notes, white papers, and user guides providing more detailed explanations of how to effectively design with MegaCore functions and the Quartus II software are available at the Altera web site (www.altera.com). Refer also to the following references:

- Hogenauer, Eugene. *An Economical Class of Digital Filters For Decimation and Interpolation*, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-29, pp. 155-162, April 1981.
- U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 2nd Edition, Springer, 2004.
- *MegaCore IP Library Release Notes and Errata*.
- *Altera Software Installation and Licensing* manual.
- *AN320: OpenCore Plus Evaluation of Megafunctions*.
- *DSP Builder User Guide*.
- *Avalon Interface Specifications*.
- *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.
- *AN442: Tool Flow Design of Digital IF for Wireless Systems*.
- *AN455: Understanding CIC Compensation Filters*.

Revision History

The following table displays the revision history for this user guide.

Date	Version	Changes Made
May 2011	11.0	<ul style="list-style-type: none"> Updated support level to final support for Arria[®] II GX, Arria II GZ, Cyclone[®] III LS, and Cyclone IV GX devices. Updated support level to HardCopy Compilation for HardCopy III, HardCopy IV E, and HardCopy IV GX devices.
December 2010	10.1	<ul style="list-style-type: none"> Added preliminary support for Arria II GZ devices. Updated support level to final support for Stratix[®] IV GT devices.
July 2010	10.0	<ul style="list-style-type: none"> Added preliminary support for Stratix V devices Updated parameter name Rate factor to Rate change factor.
November 2009	9.1	<ul style="list-style-type: none"> Maintenance update Preliminary support for Cyclone III LS, Cyclone IV, and HardCopy IV GX devices
March 2009	9.0	<ul style="list-style-type: none"> Added an option to optimize for speed Preliminary support for Arria II GX
November 2008	8.1	<ul style="list-style-type: none"> Full support for Stratix III Applied new technical publications style Withdrawn support for UNIX
May 2008	8.0	<ul style="list-style-type: none"> Full support for Cyclone III Preliminary support for Stratix IV
October 2007	7.2	<ul style="list-style-type: none"> Full support for Arria GX
May 2007	7.1	<ul style="list-style-type: none"> Added description of new features for variable interpolation/decimation rate and compensation filter coefficients generation Preliminary support for Arria[™] GX Full support for Stratix II GX and HardCopy[®] II devices
December 2006	7.0	<ul style="list-style-type: none"> Preliminary support for Cyclone III
December 2006	6.1	<ul style="list-style-type: none"> First release of this user guide

How to Contact Altera

For the most up-to-date information about Altera[®] products, refer to the following table.

Contact 1	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature






Contact 1	Contact Method	Address
Non-technical support (General) (Software Licensing)	Email	nacomp@altera.com
	Email	authorization@altera.com

Note to table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

This document uses the typographic conventions shown in the following table.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, Save As dialog box.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, and software utility names. For example, \qdesigns directory, d: drive, and chiptrip.gdf file.
<i>Italic Type with Initial Capital Letters</i>	Indicates document titles. For example: <i>AN 519: Stratix IV Design Guidelines</i> .
<i>Italic type</i>	Indicates variables. For example, $n + 1$. Variable names are enclosed in angle brackets (< >). For example, <file name> and <project name>. .pof file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
“Subheading Title”	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”
Courier type	Indicates signal, port, register, bit, block, and primitive names. For example, data1, tdi, and input. Active-low signals are denoted by suffix n. Example: resetn. Indicates command line commands and anything that must be typed exactly as it appears. For example, c:\qdesigns\tutorial\chiptrip.gdf. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword SUBDESIGN), and logic function names (for example, TRI).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
 CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
 WARNING	A warning calls attention to a condition or possible situation that can cause you injury.
	The angled arrow instructs you to press the enter key.
	The feet direct you to more information about a particular topic.